Learning Optimal Power Flow: Worst-Case Guarantees for Neural Networks

Andreas Venzke*, Guannan Qu†, Steven Low† and Spyros Chatzivasileiadis*
*Department of Electrical Engineering, Technical University of Denmark (DTU), Kgs. Lyngby, Denmark
†Department of Computing and Mathematical Sciences, California Institute of Technology, Pasadena, CA 91125, USA
E-mail: {andven, spchatz}@elektro.dtu.dk, {gqu, slow}@caltech.edu

***Abstract*—This paper introduces for the first time a framework to obtain provable worst-case guarantees for neural network performance, using learning for optimal power flow (OPF) problems as a guiding example. Neural networks have the potential to substantially reduce the computing time of OPF solutions. However, the lack of guarantees for their worst-case performance remains a major barrier for their adoption in practice. This work aims to remove this barrier. We formulate mixed-integer linear programs to obtain worst-case guarantees for neural network predictions related to (i) maximum constraint violations, (ii) maximum distances between predicted and optimal decision variables, and (iii) maximum sub-optimality. We demonstrate our methods on a range of PGLib-OPF networks up to 300 buses. We show that the worst-case guarantees can be up to one order of magnitude larger than the empirical lower bounds calculated with conventional methods. More importantly, we show that the worst-case predictions appear at the boundaries of the training input domain, and we demonstrate how we can systematically reduce the worst-case guarantees by training on a larger input domain than the domain they are evaluated on.**

***Index Terms*—Neural networks, mixed-integer linear programming, optimal power flow.**

I. Introduction

The optimal power flow (OPF) problem is an essential tool for electricity markets, for power system operation, and planning [1]. In its standard form, the OPF minimizes an objective function (e.g. generation cost) subject to the power flow equations and the operational constraints (e.g. line limits). As the non-linear AC power flow equations render the AC-OPF problem non-convex [2], the linear DC-OPF approximation is often used instead [3]. The substantial increase of uncertainty in generation and demand requires to solve OPF repeatedly and closer to real-time, in order to analyze a large number of scenarios; this leads to significant computational challenges [4]. Neural networks present a promising alternative to conventional optimization solvers, achieving a speed-up of several orders of magnitude [5]–[9]. However, the lack of any guarantees related to the neural network performance presents a major barrier towards their application in safety-critical systems. In this work, we introduce for the first time a framework to obtain worst-case guarantees for neural networks which predict solutions to DC-OPF problems.

The work of A. Venzke was carried out while visiting the Department of Computing and Mathematical Sciences at the California Institute of Technology, Pasadena, CA 91125, USA. The work of A. Venzke and S. Chatzivasileiadis is supported by the multiDC project, funded by Innovation Fund Denmark, Grant Agreement No. 6154-00020.

Machine learning including neural networks have been applied to a range of power system applications over the past three decades; for a recent survey please refer to [10]. The focus of this work is on obtaining guarantees for machine learning approaches such as the ones in [5]–[9], which predict solutions to OPF problems and replace the use of conventional optimization solvers. These approaches can result to larger computational speed-ups compared to predicting inactive constraints [11] or warm-start points [12] that could accelerate conventional optimization solvers. The work in [5] trains neural networks to directly predict the solution to DC-OPF problems, achieving a speed-up of two orders of magnitude (i.e., 100 times faster). The same authors extend this framework to include security constraints in [6]. The work in [13] proposes an off-line algorithm to identify the sets of active constraints and, based on these, directly computes solutions to DC-OPF problems on-line. The work in [7] extends this approach to neural networks predicting the active set. The work in [8] demonstrates that both the approaches in [5] and [13] can fail to predict feasible solutions, i.e., solutions satisfying the power system constraints, and proposes an alternative training procedure to improve the feasibility of the obtained predictions. Using neural networks, the work in [9] directly predicts solutions to AC-OPF problems and relies on a penalization of constraint violations during training to improve feasibility.

While the works [5]–[9] report substantial computational speed-ups and empirically analyse accuracy and feasibility, *no guarantees* for the neural network performance are provided. By evaluating the worst-case performance on the discrete samples for the entire training and test dataset only an empirical lower bound of the worst-case guarantee can be obtained. To the best of our knowledge, this work is the first to introduce a framework that obtains *exact* worst-case guarantees over the *entire* input domain, for neural networks predicting solutions to DC-OPF problems. To this end, we leverage recent advancements in evaluating the adversarial robustness of neural networks using mixed-integer programming [14]–[16]. Our previous work [16] focused on power system security assessment and provided performance guarantees for classification neural networks. While these works [14]–[16] focus on obtaining *local* robustness certificates that no adversarial examples exist (i.e., input perturbations around a given sample which lead to a wrong classification), in this work we

introduce a framework to obtain *global* worst-case guarantees over the entire input domain. The main contributions of our work are:

1) We introduce a framework to compute worst-case guarantees for (i) physical constraint violations, (ii) maximum distance between predicted and optimal decision variables, and (iii) sub-optimality, leveraging mixed-integer linear reformulations of neural networks.
2) We demonstrate our methodology on PGLib-OPF networks of up to 300 buses. We show (i) that the worst-case *guarantees over the entire input domain* can be up to an order of magnitude larger than the empirical lower bounds obtained with conventional methods; and (ii) that these worst-case guarantees can be systematically reduced by training on a larger input domain than the domain these neural networks are evaluated on.

The structure of this paper is as follows: In Section II, we formulate the DC-OPF and its KarushKuhnTucker (KKT) conditions, and explain the neural network architecture and training to predict solutions to DC-OPF problems. In Section III, leveraging mixed-integer reformulations of neural networks, we introduce the framework to compute worst-case guarantees. Section IV demonstrates our methodology on a range of PGLib-OPF networks. Section V concludes. The code to reproduce all simulation results is available online [17].

II. Learning DC-OPF with Neural Networks

First, we state the DC-OPF problem and its KKT conditions (which we will use at a later stage in Section III-C), and then we detail the architecture and training process of neural networks predicting solutions to DC-OPF problems.

A. DC Optimal Power Flow (DC-OPF) Formulation

An electric power grid consists of an n_b number of buses (denoted with the set $\mathcal{N}$) and an n_line number of lines (denoted with the set $\mathcal{L}$). Each line connects a bus $i \in \mathcal{N}$ to another bus $j \in \mathcal{N}$, $(i,j) \in \mathcal{L}$. Set $\mathcal{G}$ (a subset of $\mathcal{N}$) collects the n_g number of buses that have a generator connected to them. The vector $\mathbf{p}_\text{g}$ of size n_g denotes the generator active power output and the matrix $\mathbf{M}_\text{g}$ of size $n_\text{b} \times n_\text{g}$ maps the generators to the buses. A number n_d of buses has a load connected to them. The vector $\mathbf{p}_\text{d}$ of size n_d denotes the active power demands and the matrix $\mathbf{M}_\text{d}$ of size $n_\text{b} \times n_\text{d}$ maps the loads to the buses. In the DC-OPF formulation, the voltage magnitudes are assumed to be constant at all buses, and only the voltage angles $\boldsymbol{\theta}$ of size n_b are included as variables. The DC-OPF problem can be formulated as:

$$\min_{\mathbf{p}_\text{g}, \boldsymbol{\theta}} \quad \mathbf{c}^T \mathbf{p}_\text{g} \tag{1}$$

$$\text{s.t.} \quad \mathbf{M}_\text{g}\mathbf{p}_\text{g} - \mathbf{M}_\text{d}\mathbf{p}_\text{d} = \mathbf{B}_\text{bus}\boldsymbol{\theta} \quad : \boldsymbol{\lambda} \tag{2}$$

$$\mathbf{p}_\text{line}^\text{min} \leq \mathbf{B}_\text{line}\boldsymbol{\theta} \leq \mathbf{p}_\text{line}^\text{max} \quad : \boldsymbol{\mu}_\text{line}^\text{min}, \boldsymbol{\mu}_\text{line}^\text{max} \tag{3}$$

$$\mathbf{p}_\text{g}^\text{min} \leq \mathbf{p}_\text{g} \leq \mathbf{p}_\text{g}^\text{max} \quad : \boldsymbol{\mu}_\text{g}^\text{min}, \boldsymbol{\mu}_\text{g}^\text{max} \tag{4}$$

The objective function in (1) minimizes the generation cost, with a positive unique linear cost term $\mathbf{c}$ associated to each generator output. The nodal power balance in (2) ensures that the power generation, power demand and in- and out-going flows are balanced at each bus. The term $\mathbf{B}_\text{bus}$ defines the bus admittance matrix, and the term $\mathbf{B}_\text{line}$ the line admittance matrix. For brevity, refer to [3] for the full details. The active power line flows in (3) are a function of the line admittance matrix $\mathbf{B}_\text{line}$ and the voltage angles $\boldsymbol{\theta}$. We fix the voltage angle corresponding to the slack bus $\boldsymbol{\theta}^\text{slack} = 0$ to remove the trivial non-uniqueness of the obtained DC-OPF solution due to the singularity of the bus admittance matrix $\mathbf{B}_\text{bus}$. The physical constraints comprise minimum and maximum limits on the active line flow in (3) and the active power generation in (4), respectively. Each constraint is associated with a dual variable, denoted with $\boldsymbol{\lambda}$ for equality constraints and $\boldsymbol{\mu}$ for inequality constraints. The KKT conditions of the DC-OPF problem in (1)–(4) can be written as:

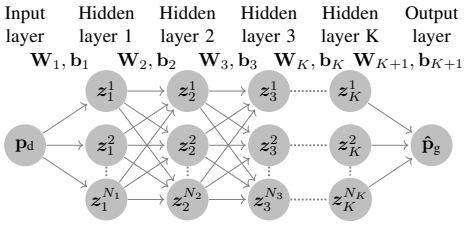


Fig. 1. Illustration of the neural network architecture to predict the mapping from the active power demand $\mathbf{p}_\text{d}$ to the optimal generation $\hat{\mathbf{p}}_\text{g}$: The neural network consists of K hidden layers with N_k neurons each with $k = 1, ..., K$. At each neuron of the hidden layers, a ReLU activation function is applied.

$$\mathbf{c} - \boldsymbol{\mu}_\text{g}^\text{min} + \boldsymbol{\mu}_\text{g}^\text{max} + \mathbf{M}_\text{g}^T\boldsymbol{\lambda} = 0 \tag{5}$$

$$-\mathbf{B}_\text{line}^\mathbf{T}\boldsymbol{\mu}_\text{line}^\text{min} + \mathbf{B}_\text{line}^\mathbf{T}\boldsymbol{\mu}_\text{line}^\text{max} - \mathbf{B}_\text{bus}\boldsymbol{\lambda} = 0 \tag{6}$$

$$\boldsymbol{\mu}_\text{line}^\text{min}(\mathbf{p}_\text{line}^\text{min} - \mathbf{B}_\text{line}\boldsymbol{\theta}) = 0,\ \boldsymbol{\mu}_\text{line}^\text{min}(\mathbf{B}_\text{line}\boldsymbol{\theta} - \mathbf{p}_\text{line}^\text{max}) = 0 \tag{7}$$

$$\boldsymbol{\mu}_\text{g}^\text{min}(\mathbf{p}_\text{g}^\text{min} - \mathbf{p}_\text{g}) = 0,\ \boldsymbol{\mu}_\text{g}^\text{max}(\mathbf{p}_\text{g} - \mathbf{p}_\text{g}^\text{max}) = 0 \tag{8}$$

$$\boldsymbol{\mu}_\text{g}^\text{min} \geq 0,\ \boldsymbol{\mu}_\text{g}^\text{max} \geq 0, \boldsymbol{\mu}_\text{line}^\text{min} \geq 0,\ \boldsymbol{\mu}_\text{line}^\text{max} \geq 0 \tag{9}$$

$$(2) - (4) \tag{10}$$

The stationarity conditions are described in (5) and (6). The complementary slackness conditions are enforced in (7) and (8). The primal and dual feasibility corresponds to (10) and (9), respectively. As the DC-OPF in (1)–(4) is a linear program, satisfying the KKT conditions is necessary and sufficient for optimality [18], given the DC-OPF problem is feasible.

B. Neural Network Architecture and Training

This subsection details the neural network architecture and the training procedure in order to learn the mapping between an instance of the power demand $\mathbf{p}_\text{d}$ and the associated optimal generation dispatch $\mathbf{p}_\text{g}$ of the DC-OPF, see also (1)–(4). We assume the power system topology is fixed, i.e., $\mathbf{B}_\text{bus}$ and $\mathbf{B}_\text{line}$ remain constant, and the load domain $\mathbf{p}_\text{d} \in \mathcal{D}$ is restricted to a convex polytope characterized by matrix $\mathbf{A}_\text{d}$ and vector $\mathbf{b}_\text{d}$:

$$\mathbf{A}_\text{d}\mathbf{p}_\text{d} \leq \mathbf{b}_\text{d} \tag{11}$$

On this load domain, we assume that the DC-OPF mapping from system demand $\mathbf{p}_\text{d}$ to the optimal generation dispatch $\mathbf{p}_\text{g}$

is unique, i.e., a singleton. It is shown in [19] that the DC-OPF solution is unique almost surely in an appropriate space. This is a requirement for the learning task as the neural network predicts one optimal generation dispatch $\hat{\mathbf{p}}_\mathrm{g}$ for the active power demand input. The neural network architecture to learn this mapping is illustrated in Fig. 1. The architecture comprises a number K of fully-connected hidden layers, each of which consists of a number of neurons N_k with $k = 1, ..., K$. The neural network input vector is the active power demand $\mathbf{p}_\mathrm{d}$ and the output vector is the prediction of the optimal generation dispatch $\hat{\mathbf{p}}_\mathrm{g}$. Note that the entry of $\hat{\mathbf{p}}_\mathrm{g}$ corresponding to the slack bus $(\hat{\mathbf{p}}_\mathrm{g})^\mathrm{slack}$ is not predicted by the neural network as it is not an independent variable. The slack bus generation is defined by the difference in predicted generation and demand:

$$(\hat{\mathbf{p}}_\mathrm{g})^\mathrm{slack} = \sum_{i \in \mathcal{N}} (\mathbf{M}_\mathrm{d} \mathbf{p}_\mathrm{d})^i - \sum_{i \in \mathcal{G} \setminus \mathrm{slack}} (\hat{\mathbf{p}}_\mathrm{g})^i \tag{12}$$

The superscripts are used to denote the corresponding entries of the vectors. The input to the first and subsequent hidden layers $\hat{\mathbf{z}}_k$ of the neural network is defined as:

$$\hat{\mathbf{z}}_1 = \mathbf{W}_1 \mathbf{p}_\mathrm{d} + \mathbf{b}_1 \tag{13}$$

$$\hat{\mathbf{z}}_{k+1} = \mathbf{W}_{k+1} \mathbf{z}_k + \mathbf{b}_{k+1} \quad \forall k = 1, ..., K-1 \tag{14}$$

The weight matrices $\mathbf{W}_k$ have dimensions $N_{k+1} \times N_k$ and the bias vector $\mathbf{b}$ has dimension N_{k+1}. Each neuron in the hidden layer applies a non-linear activation function to the input. In the following, we use the ReLU activation function, which is used by the majority of neural network applications in recent years, as it has been found to accelerate neural network training [20]:

$$\mathbf{z}_k^i = \max(\hat{\mathbf{z}}_k^i, 0) \quad \forall k = 1, ..., K \quad \forall i = 1, ..., N_k \tag{15}$$

The ReLU activation function in (15) outputs 0 if the input is negative, otherwise it propagates the input. Note that the $\max$ operator is applied element-wise to the vector $\hat{\mathbf{z}}_k$. The predicted generator dispatch of the neural network can be evaluated as follows:

$$(\hat{\mathbf{p}}_\mathrm{g})^\mathrm{nsg} = \mathbf{W}_{K+1} \mathbf{z}_K + \mathbf{b}_{K+1} \tag{16}$$

The term $(\hat{\mathbf{p}}_\mathrm{g})^\mathrm{nsg}$ denotes the $n_\mathrm{g} - 1$ entries of $\hat{\mathbf{p}}_\mathrm{g}$ that do not correspond to the slack bus. To train neural networks, the first step is to create a dataset of demand instances $\mathbf{p}_\mathrm{d} \in \mathcal{D}$ and their corresponding optimal generation $\mathbf{p}_\mathrm{g}$ by e.g. using historical data and simulation tools. The obtained dataset is split into a training and test set. Then, during neural network training, the weight matrices $\mathbf{W}$ and biases $\mathbf{b}$ are optimized using stochastic gradient descent to minimize a loss function, e.g. the mean squared error between the prediction $\hat{\mathbf{p}}_\mathrm{g}$ and the training dataset $\mathbf{p}_\mathrm{g}$. In previous works (e.g. [5], [8]), the performance of the trained neural network is evaluated on the test set using *statistical* metrics such as accuracy or share of feasible instances. This procedure does not provide any *guarantees* related to the worst-case performance of the trained neural network over the entire input domain $\mathbf{p}_\mathrm{d} \in \mathcal{D}$.

III. Worst-Case Guarantees for Neural Networks

We first state the mixed-integer reformulation of trained neural networks and address issues related to scalability. Then, we introduce our framework to compute worst-case guarantees.

A. Mixed-Integer Reformulation of Trained Neural Networks

To include the trained neural network equations in an optimization framework, we follow the work in [15] and reformulate the maximum operator in the ReLU activations (15) using binary variables $\mathbf{b}_k \in \{0,1\}^{N_k}$ for all $k = 1, ..., K$ and suitable minimum and maximum bounds on the neuron output $\hat{\mathbf{z}}^\mathrm{min}$ and $\hat{\mathbf{z}}^\mathrm{max}$:

$$\mathbf{z}_k^i \leq \hat{\mathbf{z}}_k^i - \hat{\mathbf{z}}_k^{\mathrm{min},i}(1 - \mathbf{b}_k^i) \; \forall k = 1, ..., K \, \forall i = 1, ..., N_k \tag{17}$$

$$\mathbf{z}_k^i \geq \hat{\mathbf{z}}_k^i \qquad \forall k = 1, ..., K \, \forall i = 1, ..., N_k \tag{18}$$

$$\mathbf{z}_k^i \leq \hat{\mathbf{z}}_k^{\mathrm{max},i} \mathbf{b}_k^i \qquad \forall k = 1, ..., K \, \forall i = 1, ..., N_k \tag{19}$$

$$\mathbf{z}_k^i \geq \mathbf{0} \qquad \forall k = 1, ..., K \; \forall i = 1, ..., N_k \tag{20}$$

$$\mathbf{b}_k \in \{0,1\}^{N_k} \qquad \forall k = 1, ..., K \tag{21}$$

Observe that $\hat{\mathbf{z}}_k^i$ refers to the neuron (ReLU) input and $\mathbf{z}_k^i$ to the neuron (ReLU) output. Note that the minimum and maximum bounds on the neuron output $\hat{\mathbf{z}}^\mathrm{min}$ and $\hat{\mathbf{z}}^\mathrm{max}$ have to be chosen large enough to not be binding and as small as possible to facilitate tight bounds for the branch-and-bound algorithm. In case the input to the i-th neuron in layer k is $\hat{\mathbf{z}}_k^i \leq 0$ then the corresponding binary variable $\mathbf{b}_k^i$ is 0 and (19) and (20) constrain the neuron output $\mathbf{z}_k^i$ to 0. The constraints in (17) and (18) are non-binding if $\hat{\mathbf{z}}_k^i < 0$ holds. If the input to the neuron is $\hat{\mathbf{z}}_k^i \geq 0$, then the binary variable is 1 and (17) and (18) constrain the neuron output $\mathbf{z}_k^i$ to the input $\hat{\mathbf{z}}_k^i$. The constraints in (19) and (20) are non-binding if $\hat{\mathbf{z}}_k^i > 0$ holds.

As this reformulation introduces one binary variable for each neuron in the hidden layers, we use a combination of the works in [15] and [14] and employ three techniques to maintain scalability of the resulting mixed-integer linear programs (MILPs). First, we sparsify the weight matrices $\mathbf{W}$ during training, i.e., we gradually enforce a defined share of entries to be zero. Second, we apply the concept of ReLU stability [14]: All neurons for which the activation is always active or always inactive on both the training and test set are fixed to this status in the MILP reformulation, and the corresponding binaries are eliminated. Third, we use several techniques to compute increasingly tighter bounds $\hat{\mathbf{z}}^\mathrm{min}$ and $\hat{\mathbf{z}}^\mathrm{max}$. We initialize the bounds using interval arithmetic (for details see [15]). Then, to compute tighter bounds, we minimize and maximize the output of each neuron $\mathbf{z}_k^i$ subject to the linear relaxation of the trained neural network (13), (14), (16)–(21), and subject to the restricted input domain in (11). Note that for the linear relaxation only we relax the binary variables $\mathbf{b}_k$ to continuous variables between $\mathbf{0}$ and $\mathbf{1}$. Finally, we repeat this step using the full MILP formulation of the trained neural networks. As a result, we obtain tightened bounds $\hat{\mathbf{z}}^\mathrm{min}$ and $\hat{\mathbf{z}}^\mathrm{max}$ for the branch-and-bound algorithm. Note that in the following, when solving MILPs to obtain worst-case guarantees, we always solve the full MILP formulation and do not use a relaxation.

B. Worst-Case Guarantees for Constraint Violations

The mixed-integer reformulation of trained neural networks allows us to formulate optimization problems to obtain worst-case guarantees for the physical constraint violation. We define the maximum violation of the constraints on active generator power ν_{g} in (4) and on active line flows ν_{line} in (3) as:

$$\nu_{\text{g}} = \max(\hat{\mathbf{p}}_{\text{g}} - \mathbf{p}_{\text{g}}^{\text{max}}, \mathbf{p}_{\text{g}}^{\text{min}} - \hat{\mathbf{p}}_{\text{g}}, \mathbf{0}) \tag{22}$$

$$\nu_{\text{line}} = \max(|\mathbf{B}_{\text{line}}\tilde{\mathbf{B}}_{\text{bus}}^{-1}(\mathbf{M}_{\text{g}}\hat{\mathbf{p}}_{\text{g}} - \mathbf{M}_{\text{d}}\mathbf{p}_{\text{d}})^{\text{nsb}}| - \mathbf{p}_{\text{line}}^{\text{max}}, \mathbf{0}) \tag{23}$$

The term 'nsb' denotes all buses except the slack bus. To compute the maximum constraint violation of the line flow in (23), we compute the line flow based on the neural network prediction $\hat{\mathbf{p}}_{\text{g}}$ and system loading $\mathbf{p}_{\text{d}}$. To this end, we remove the column and row from the bus admittance matrix and invert the resulting reduced bus admittance matrix $\tilde{\mathbf{B}}_{\text{bus}}$, inserting (2) in (3). Note that the product $\mathbf{B}_{\text{line}}\tilde{\mathbf{B}}_{\text{bus}}^{-1}$ is the well-known "Power Transfer Distribution Factors" (PTDF) matrix; please refer to [21] for more details. In both (22) and (23), we take the overall non-negative maximum over the violations. Note that we take the absolute value $|\cdot|$ of the line flow in (23). In previous works, these metrics have only been evaluated empirically on the datasets. Here, to compute the worst-case generator constraint violation for the entire defined input domain, we solve:

$$\max_{\hat{\mathbf{p}}_{\text{g}}, \mathbf{p}_{\text{d}}, \mathbf{b}, \mathbf{z}, \hat{\mathbf{z}}, \nu_{\text{g}}} \quad \nu_{\text{g}} \tag{24}$$

$$\text{s.t. } (11) - (14), (16), (17) - (21), (22) \tag{25}$$

Similarly, to compute the maximum line constraint violation ν_{line}, we maximize ν_{line} subject to (25), replacing (22) with (23). As the input domain in (11) is a convex polytope and we reformulate the max-operators in (22) and (23) using integer variables, the optimization problem (24)–(25) can be cast as MILP. If the MILP is solved to zero MILP gap, i.e., to global optimality, then the bound is exact, and we obtain the provable guarantee that no input $\mathbf{p}_{\text{d}} \in \mathcal{D}$ to the neural network exist which will lead to constraint violations larger than the obtained values of ν_{g} and ν_{line}. At the same time, the obtained values of $\mathbf{p}_{\text{d}}$ are the neural network inputs which lead to the maximum constraint violations. If the MILP is solved to a non-zero optimality gap, then we obtain an upper bound on the worst-case violations ν_{g} and ν_{line}. If, additionally, the MILP solver identifies a feasible solution, then the values of ν_{g} and ν_{line} corresponding to the feasible solution serve as a lower bound on the worst-case violations. Note that in the simulation results in Section IV, we solve all MILPs to zero optimality gap.

C. Worst-Case Guarantees for Distance of Predicted to Optimal Decision Variables and for Sub-Optimality

In the following, we formulate optimization problems to obtain (i) worst-case guarantees for the maximum distance between the predicted and the optimal decision variables ν_{dist} and (ii) worst-case guarantees for the sub-optimality of the cost function ν_{opt} resulting from the predicted solution:

$$\nu_{\text{dist}} = \max(|\tfrac{|\hat{\mathbf{p}}_{\text{g}} - \mathbf{p}_{\text{g}}|}{\mathbf{p}_{\text{g}}^{\text{max}} - \mathbf{p}_{\text{g}}^{\text{min}}}|) \tag{26}$$

$$\nu_{\text{opt}} = \mathbf{c}^T(\hat{\mathbf{p}}_{\text{g}} - \mathbf{p}_{\text{g}}) \tag{27}$$

The term $\mathbf{p}_{\text{g}}$ denotes the optimal solution to the DC-OPF problem for a given input loading $\mathbf{p}_{\text{d}}$. We normalize the distance ν_{dist} element-wise by the corresponding generator limits and compute the maximum over all generator set-points. The distance ν_{dist} characterizes for the entire input domain the largest mismatch of all generator set-points between the prediction of the neural network and the ground-truth DC-OPF solution. We formulate the following bi-level problem to compute the worst-case distance ν_{dist}:

$$\max_{\hat{\mathbf{p}}_{\text{g}}, \mathbf{p}_{\text{g}}, \mathbf{p}_{\text{d}}, \mathbf{b}, \mathbf{z}, \hat{\mathbf{z}}, \nu_{\text{dist}}} \quad \nu_{\text{dist}} \tag{28}$$

$$\text{s.t. } (11) - (14), (16), (17) - (21), (26) \tag{29}$$

$$\mathbf{p}_{\text{g}} \in \arg\min_{\boldsymbol{p}_{\text{g}}, \boldsymbol{\theta}} \{(1) \text{ s.t. } (2) - (4)\} \tag{30}$$

The lower-level comprises the DC-OPF formulation and defines the optimal generation $\mathbf{p}_{\text{g}}$ as a function of the load input $\mathbf{p}_{\text{d}}$. The upper-level problem maximizes the distance of the predicted to the optimal solution of the DC-OPF for the defined load input domain. We replace the lower-level problem with its KKT conditions and rewrite the optimization problem:

$$\max_{\hat{\mathbf{p}}_{\text{g}}, \mathbf{p}_{\text{g}}, \mathbf{p}_{\text{d}}, \mathbf{b}, \mathbf{z}, \hat{\mathbf{z}}, \nu_{\text{dist}}, \boldsymbol{\theta}, \boldsymbol{\lambda}, \boldsymbol{\mu}} \quad \nu_{\text{dist}} \tag{31}$$

$$\text{s.t. } (29), (5) - (10) \tag{32}$$

By maximizing ν_{opt} in the the objective function and replacing (26) with (27) we can compute worst-case guarantees for the sub-optimality of the predicted solution. To achieve tractability of this formulation, we reformulate the non-linear complementary slackness conditions (7) and (8) in (32) using the Fortuny-Amat McCarl linearization [22]:

$$\mathbf{p}_{\text{line}}^{\text{min}} - \mathbf{B}_{\text{line}}\boldsymbol{\theta} \geq -\mathbf{r}_{\text{line}}^{\text{min}}\mathbf{M}_{\text{line}}^{\text{min}}, \ \boldsymbol{\mu}_{\text{line}}^{\text{min}} \leq (1 - \mathbf{r}_{\text{line}}^{\text{min}})\mathbf{M}_{\text{line}}^{\text{min}} \tag{33}$$

$$\mathbf{B}_{\text{line}}\boldsymbol{\theta} - \mathbf{p}_{\text{line}}^{\text{max}} \geq -\mathbf{r}_{\text{line}}^{\text{max}}\mathbf{M}_{\text{line}}^{\text{max}}, \ \boldsymbol{\mu}_{\text{line}}^{\text{max}} \leq (1 - \mathbf{r}_{\text{line}}^{\text{max}})\mathbf{M}_{\text{line}}^{\text{max}} \tag{34}$$

$$\mathbf{p}_{\text{g}}^{\text{min}} - \mathbf{p}_{\text{g}} \geq -\mathbf{r}_{\text{g}}^{\text{min}}\mathbf{M}_{\text{g}}^{\text{min}}, \ \boldsymbol{\mu}_{\text{g}}^{\text{min}} \leq (1 - \mathbf{r}_{\text{g}}^{\text{min}})\mathbf{M}_{\text{g}}^{\text{min}} \tag{35}$$

$$\mathbf{p}_{\text{g}} - \mathbf{p}_{\text{g}}^{\text{max}} \geq -\mathbf{r}_{\text{g}}^{\text{max}}\mathbf{M}_{\text{g}}^{\text{max}}, \ \boldsymbol{\mu}_{\text{g}}^{\text{max}} \leq (1 - \mathbf{r}_{\text{g}}^{\text{max}})\mathbf{M}_{\text{g}}^{\text{max}} \tag{36}$$

This models the complementary slackness conditions with one binary variable $\mathbf{r}$ and one large non-binding constant $\boldsymbol{M}$ for each condition. Note that the constant $\boldsymbol{M}$ has to be chosen sufficiently large to not be binding, while at the same time small enough to maintain numerical well-conditioning of the mixed-integer program. For details on bi-level programming and this reformulation, please refer to [23]. The resulting optimization problem is a MILP which includes integer variables related to the reformulation of the neural network, and related to the reformulation of the lower-level problem. If this MILP is solved to zero MILP gap (and if constraint qualifications for global optimality to the bi-level problem are satisfied [23]), the bound is exact and we obtain the provable guarantee that no input $\mathbf{p}_{\text{d}} \in \mathcal{D}$ exist with distances or sub-optimality larger than the obtained values of ν_{dist} and ν_{opt}.

TABLE I
TEST CASE CHARACTERISTICS

Test cases	n_{d}	n_{g}	n_{b}	n_{line}	Max. loading
case9	3	3	9	9	315.0 MW
case30	21	2	30	41	283.4 MW
case39	21	10	39	46	6254.2 MW
case57	42	4	57	80	1250.8 MW
case118	99	19	118	186	4242.0 MW
case162	113	12	162	284	7239.1 MW
case300	199	57	300	411	23525.9 MW

TABLE II
PERFORMANCE AVERAGED OVER TEST SET SAMPLES

Test cases	MAE (%)	ν_{g} (MW)	ν_{line} (MW)	ν_{dist} (%)	ν_{opt} (%)
case9	0.04	0.07	0.02	0.06	0.04
case30	0.03	0.00	0.01	0.03	-0.00
case39	0.07	0.71	1.02	0.30	0.00
case57	0.01	0.24	0.00	0.03	-0.01
case118	0.31	8.21	1.35	3.35	0.00
case162	0.61	9.11	2.07	4.08	0.01
case300	0.90	15.33	96.13	18.01	-0.02

IV. SIMULATION & RESULTS

We demonstrate our methodology on a range of PGLib-OPF networks v19.05 of up to 300 buses from [24]. The test case characteristics are listed in Table I. The *case9* is taken from MATPOWER [25]. We assume that the input domain $\boldsymbol{p}_{\mathrm{d}} \in \mathcal{D}$ in (11) is defined as $0.6\,\boldsymbol{p}_{\mathrm{d}}^{\max} \leq \boldsymbol{p}_{\mathrm{d}} \leq 1.0\,\boldsymbol{p}_{\mathrm{d}}^{\max}$, i.e. each load can fluctuate individually from 60% to 100% of its maximum loading. Note that the maximum loading level $\boldsymbol{p}_{\mathrm{d}}^{\max}$ is defined according to [24], [25], and the sum of the maximum loading is shown in Table I. We did not consider loading levels larger than 100% as we observed that this frequently leads to infeasibility of the DC-OPF problem. This would require load shedding and represents an abnormal system situation. To create the datasets, we use Latin hypercube sampling [26], draw 100'000 samples from the input domain $\mathcal{D}$, and solve a DC-OPF for each of the samples using MATPOWER [25] to generate the corresponding optimal solutions. Out of these input-output pairs we use 80% for training and 20% for testing.

The neural network architecture comprises three hidden layers with 50 neurons each. As we will demonstrate (and has also been shown in [8]), the size of this architecture is sufficient to obtain low generalization errors of the neural networks on the unseen test set. As described in Section III-A, we sparsify the neural network during training by gradually setting the smallest weight entries to zero until 80% of the weight entries are zero; that means that only 20% of weight entries are non-zero at the end of training. We use TensorFlow [27] for neural network training with the following specifications. During training, we minimize the mean squared error between the neural network prediction and the true optimal solutions. We define the maximum number of training epochs to 250 and split the dataset into 2000 batches. We use early stopping and we recover the neural network weights and biases that achieved the lowest generalization error on the test set. As the neural network training is highly non-linear, we repeat the training and evaluation process 5 times, and report averaged values for all simulation results. We formulate the MILPs in YALMIP [28] and solve them using Gurobi. For the Fortuny-Amat McCarl linearization in (33)–(36) we choose all constants $\boldsymbol{M}$ to be 10^5. After solving the MILPs, we verify that the complementary slackness conditions are satisfied and the constants are non-binding. All computational experiments are carried out on a laptop with i7-7820HQ CPU @2.90 GHz, 32 GB RAM and GeForce 940MX GPU. The code to reproduce all simulation results is available online [17].

A. Neural Network Performance

In the following, we evaluate the performance of the trained neural networks with four metrics: The maximum generator and line constraint violations ν_{g}, ν_{line} defined in (22) and (23), the distance of the predicted to the optimal decision variables ν_{dist} defined in (26),and the sub-optimality ν_{opt} defined in (27). Note that we normalize the sub-optimality with respect to the generation cost of the 100% loaded system state.

1) Performance Averaged over Test Set Samples: In Table II, we show the performance of the trained neural networks averaged over the unseen test dataset samples. The mean absolute error (MAE) of the predicted generation dispatch evaluates to less than 1% (normalized by the generator limits as in (26)), indicating satisfactory generalization capability of the neural networks. The averaged largest violation of active generator and line limits are less than 0.5% of the total maximum system loading in Table I. The averaged largest distances of the predicted and optimal generator dispatch ν_{dist} are less than 1% for the first four test cases, and increases up to 18% for *case300*. Note that the latter corresponds to the maximum over the vector $\hat{\mathbf{p}}_{\mathbf{g}}$ of 57 predicted generator set-points. The averaged sub-optimality ν_{opt} of the predicted solutions is negligible. Note that the sub-optimality measure can be negative if constraints are violated. The average performance on the test set indicates satisfactory neural network performance. In the following, however, we demonstrate that the worst-case guarantees for these four metrics can be up to two orders of magnitude larger than the average performance on the test set (reported in Table I).

2) Worst-Case Guarantees for Constraint Violations: We first compute the worst-case constraint violations on the entire data set, i.e. on all training and test set samples. This serves as an empirical lower bound on the worst-case guarantees. Then, using the mixed-integer linear reformulation of the trained neural networks, we solve the MILPs in (24)–(25) to compute the corresponding worst-case guarantees. In Table III, we compare the obtained empirical lower bounds with the

TABLE III
WORST-CASE GUARANTEES FOR PHYSICAL CONSTRAINT VIOLATIONS

Test cases	Emp. lower bound		Worst-case guarantee			
	ν_{g} (MW)	ν_{line} (MW)	ν_{g} (MW)	(ratio)	ν_{line} (MW)	(ratio)
case9	2.5	1.8	2.8	1.1x	1.9	1.1x
case30	1.7	0.6	3.6	2.1x	3.1	4.9x
case39	51.9	37.2	270.6	5.2x	120.0	3.2x
case57	4.2	0.0	23.7	5.6x	0.0	–
case118	149.4	15.6	997.8	6.7x	510.8	32.7x
case162	228.0	180.0	1563.3	6.9x	974.1	5.4x
case300	474.5	692.7	3658.5	7.7x	3449.3	5.0x

TABLE IV
WORST-CASE GUARANTEES FOR (I) DISTANCE OF PREDICTED TO OPTIMAL DECISION VARIABLES AND (II) SUB-OPTIMALITY

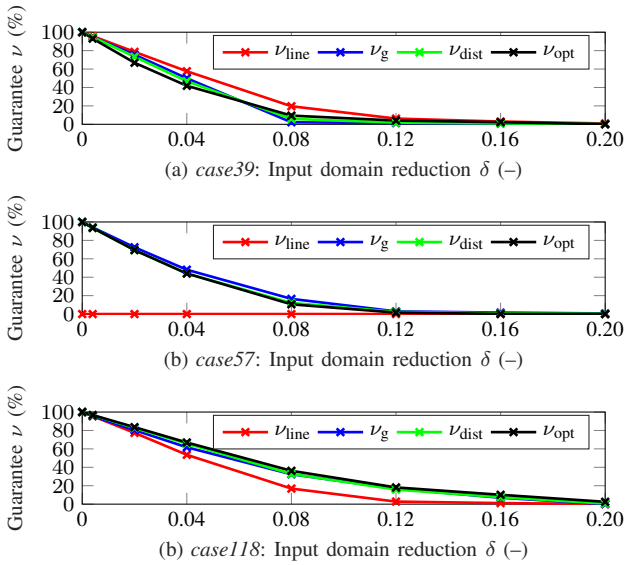
Test cases	Emp. lower bound		Worst-case guarantee			
	ν_{dist} (%)	ν_{opt} (%)	ν_{dist} (%)	(ratio)	ν_{opt} (%)	(ratio)
case9	1.2	3.3	1.4	1.2x	3.8	1.1x
case30	2.0	0.6	6.4	3.2x	2.5	3.8x
case39	6.2	0.6	64.4	10.4x	3.1	4.9x
case57	0.5	0.2	18.6	37.9x	1.8	8.1x
case118	35.0	0.2	265.7	7.6x	1.6	6.5x

worst-case guarantees related to the violation of the generator constraints ν_{g} and of the transmission line constraints ν_{line}. First, we find that the worst-case guarantees for constraint violations can be substantial. Table III shows the violations in MW-values. In percentage, the violations are on average 8.1% and up to 23.5% (*case118*) of the maximum system loading shown in Table I for each case. Second, the worst-case guarantees are on average 6.7 times and up to 32.7 times larger than the empirical lower bounds (the empirical lower bounds are obtained by evaluating the worst-case performance on the discrete samples of the entire training and test dataset; if we only consider the test set, then the worst-case guarantees are on average 255.2 times larger than the performance shown in Table II). For the *case57* system, we obtained a certificate that no input inside the input domain exists which can lead to a violation of the line constraints. Overall, these findings highlight that by only considering the performance on the dataset, the worst-case performance can be significantly underestimated, posing a risk for real-time deployment if we do not take appropriate mitigation measures. At the same time, our framework allows to obtain a provable exact certificate on the worst-case performance of neural networks.

By analyzing the solutions, we identified that for 18 out of the 35 evaluations (5 neural networks trained for each test case), the worst-case generator violation (ν_{g}) occurs for the slack bus generator, as this generator has to compensate for the mismatch in predicted generation and load. For the line limits, the worst-case violations occurred on a line directly connected to the slack bus for 24 out of the 35 evaluations. Averaged over the 7 test cases and 5 runs for each test case, it takes 3.4 minutes to compute the tightened bounds for the mixed-integer reformulation in (17) and (19), and 1.4 minutes to solve both the MILP to zero MILP gap and compute the worst-case guarantees. Based on the activation patterns on the entire dataset, on average, 17.1% of the ReLU activations are fixed to be active and 39.4% are fixed to be inactive for solving the MILPs (as described in Section III-A about ReLU stability).

3) Worst-Case Guarantees for Distance of Predicted to Optimal Decision Variables and for Sub-Optimality: In the next step, in Table IV, for the same trained neural networks and using the same procedure as in Table III, we compare the obtained empirical lower bounds and worst-case guarantees related to (i) the maximum distance between the predicted and the optimal decision variables ν_{dist} and (ii) the sub-optimality ν_{opt}. For these two metrics, we also observe that the worst-case guarantees can be substantial; they are on average 8.5 times and up to 37.9 times larger than the empirical lower bounds which are obtained by calculating the worst-case neural network performance on the discrete dataset samples. By analyzing the solutions for the metric ν_{dist}, we identified that for 12 out of the 25 evaluations, the worst-case distance between the neural network prediction and the optimal solution occurs for the slack bus generator. For the first four test cases, on average, it takes 0.3 minutes to solve both the MILPs to zero MLIP gap. For the *case118*, the average computational time increases to 25.6 minutes to solve both the MILPs to compute ν_{dist} and ν_{opt} to zero MILP gap. Note that the computational complexity increases as the KKT conditions of the DC-OPF problem are included in (31) – (32). For the *case162* and *case300*, the MILPs could not be solved to a zero MILP gap within 3 hours. Improving the tractability using decomposition techniques and validating the satisfaction of constraint qualifications for global optimality to the bi-level program in (28) – (30) are subject of our future work [23].

4) Input Domain Reduction: In the following, we demonstrate that the worst-case guarantees can be systematically reduced by training on a larger input domain than the worst-case guarantees are evaluated on. We achieve this by reducing the input domain $\mathcal{D}$ with a term δ that can vary between 0.0 and 0.2: $(0.6+\delta)\boldsymbol{p}_{\text{d}}^{\max} \leq \boldsymbol{p}_{\text{d}} \leq (1.0-\delta)\boldsymbol{p}_{\text{d}}^{\max}$. For *case39*, *case57* and *case118*, Fig. 2 shows the worst-case guarantees as a function of the input domain reduction δ. Note that the values on y-axis are normalized to 100% with respect to the worst-case values reported in Tables III and IV for the entire initial input domain. First, we can observe that the inputs (i.e., the loading $\mathbf{p}_{\text{d}}$) which lead to the worst-case performance are at the boundary of the input domain. Second, by increasing the input domain reduction δ, the worst-case bounds can be systematically reduced (e.g., for these three test cases, by reducing each dimension by $\delta = 0.08$, we can reduce all worst-case guarantees to below 20% compared to the initial domain). This implies that to reach an acceptable worst-case



(a) *case39*: Input domain reduction δ (–)

(b) *case57*: Input domain reduction δ (–)

(b) *case118*: Input domain reduction δ (–)

Fig. 2. The worst-case guarantees are shown as a function of the input domain reduction δ for *case39*, *case57* and *case118*. Note that the values are normalized to 100% with respect to the worst-case values reported in Tables III and IV for the entire initial input domain.

performance on a specified domain, the neural network can be re-trained on a larger domain if the initial performance is not satisfactory.

V. Conclusion

This work introduces for the first time a framework to obtain worst-case guarantees for neural networks. As a guiding example, we apply it to neural networks predicting DC-OPF solutions. Our work addresses a major barrier which, if removed, would enable the application of neural networks in safety-critical systems. Leveraging mixed-integer linear reformulations of trained neural networks, we can obtain worst-case guarantees with respect to the maximum physical constraint violations, the maximum distance between the predicted and the optimal decision variables, and the maximum sub-optimality. For a range of PGLib-OPF networks up to 300 buses, we show that the obtained worst-case guarantees can be up to one order of magnitude larger than the empirical lower bounds (i.e. computing the maximum of an error metric on the discrete samples of the entire dataset). More importantly, we show that the worst-case predictions appear on the boundaries of the input domain used for training. As a result, the worst-case guarantees can be systematically reduced by training the neural network on a larger input domain, and applying it on a subdomain. Future work is directed towards robust neural network training and obtaining worst-case guarantees for predicting solutions to AC-OPF problems.

References

[1] M. B. Cain, R. P. Oneill, and A. Castillo, "History of optimal power flow and formulations," *Federal Energy Regulatory Commission*, vol. 1, pp. 1–36, 2012.

[2] D. K. Molzahn and I. A. Hiskens, "A survey of relaxations and approximations of the power flow equations," *Foundations and Trends® in Electric Energy Systems*, vol. 4, no. 1-2, pp. 1–221, 2019.

[3] B. Stott, J. Jardim, and O. Alsaç, "Dc power flow revisited," *IEEE Transactions on Power Systems*, vol. 24, no. 3, pp. 1290–1300, 2009.

[4] Y. Tang, K. Dvijotham, and S. Low, "Real-time optimal power flow," *IEEE Transactions on Smart Grid*, vol. 8, no. 6, pp. 2963–2973, 2017.

[5] X. Pan, T. Zhao, and M. Chen, "Deepopf: Deep neural network for dc optimal power flow," in *2019 SmartGridComm*. IEEE, 2019, pp. 1–6.

[6] ——, "Deepopf: A deep neural network approach for security-constrained dc optimal power flow," *preprint arXiv:1910.14448*, 2019.

[7] D. Deka and S. Misra, "Learning for dc-opf: Classifying active sets using neural nets," in *2019 IEEE Milan PowerTech*. IEEE, 2019.

[8] Y. Chen and B. Zhang, "Learning to solve network flow problems via neural decoding," *arXiv preprint arXiv:2002.04091*, 2020.

[9] F. Fioretto, T. W. Mak, and P. Van Hentenryck, "Predicting ac optimal power flows: Combining deep learning and lagrangian dual methods," *arXiv preprint arXiv:1909.10461*, 2019.

[10] L. Duchesne, E. Karangelos, and L. Wehenkel, "Recent developments in machine learning for energy systems reliability management," *Proceedings of the IEEE*, 2020.

[11] S. Pineda, J. M. Morales, and A. Jimenez-Cordero, "Data-driven screening of network constraints for unit commitment," *IEEE Transactions on Power Systems*, 2020.

[12] K. Baker, "Learning warm-start points for ac optimal power flow," in *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2019, pp. 1–6.

[13] Y. Ng, S. Misra, L. A. Roald, and S. Backhaus, "Statistical learning for dc optimal power flow," in *2018 Power Systems Computation Conference (PSCC)*. IEEE, 2018, pp. 1–7.

[14] K. Y. Xiao *et al.*, "Training for faster adversarial robustness verification via inducing reLU stability," in *International Conference on Learning Representations (ICLR 2019)*, 2019.

[15] V. Tjeng, K. Y. Xiao, and R. Tedrake, "Evaluating robustness of neural networks with mixed integer programming," in *International Conference on Learning Representations (ICLR 2019)*, 2019.

[16] A. Venzke and S. Chatzivasileiadis, "Verification of neural network behaviour: Formal guarantees for power system applications," *arXiv preprint arXiv:1910.01624*, 2019.

[17] A. Venzke, G. Qu, S. Low, and S. Chatzivasileiadis, "Supplementary data and code for "Learning optimal power flow: Worst-case guarantees for neural networks"," 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3871755

[18] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[19] F. Zhou, J. Anderson, and S. H. Low, "The optimal power flow operator: Theory and computation," *arXiv preprint arXiv:1907.02219*, 2019.

[20] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 315–323.

[21] S. Chatzivasileiadis, *Optimization in Modern Power Systems*. Lecture Notes. Tech. Univ. of Denmark. Available online: https://arxiv.org/pdf/1811.00943.pdf, 2018.

[22] J. Fortuny-Amat and B. McCarl, "A representation and economic interpretation of a two-level programming problem," *Journal of the operational Research Society*, vol. 32, no. 9, pp. 783–792, 1981.

[23] S. Dempe *et al.*, "Bilevel programming problems," *Energy Systems. Springer, Berlin*, 2015.

[24] S. Babaeinejadsarookolaee *et al.*, "The power grid library for benchmarking ac optimal power flow algorithms," *arXiv preprint arXiv:1908.02788*, 2019.

[25] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "Matpower: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Transactions on power systems*, vol. 26, no. 1, pp. 12–19, 2010.

[26] M. D. McKay, R. J. Beckman, and W. J. Conover, "Comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.

[27] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[28] J. Lofberg, "Yalmip: A toolbox for modeling and optimization in matlab," in *2004 IEEE international conference on robotics and automation (IEEE Cat. No. 04CH37508)*. IEEE, 2004, pp. 284–289.